НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ім. ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Поліно В.О.
Завідувач кафедри
_____ Олександр Коваль
« __»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Геометричне моделювання в інформаційних системах»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Додаткові засоби захисту бази метаданих реєстру інформаційних ресурсів»

Виконав (-ла):
студент (-ка) IV курсу, групи ТР-61
Поліно Вадим Олександрович _____

Керівник:
Професор кафедри АПЕПС, доктор технічних наук,
Отрох Сергій Іванович _____

Рецензент:
Доцент кафедри ТК, кандидат технічних наук,
Зенів Ірина Онуфріївна _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Геометричне моделювання в інформаційних системах

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ

на дипломну роботу студенту

Поліно Вадим Олександрович
(прізвище, ім’я, по батькові)

1. Тема роботи _Додаткові засоби захисту бази метаданих реєстру інформаційних ресурсів_____

керівник роботи_____професор, д.т.н. Отрох Сергій Іванович_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від “25” травня 2020р. № **1268-с**

2. Строк подання студентом роботи _____10 червня 2020_____

3. Вихідні дані до роботи роботи: персональний комп’ютер під керуванням операційної системи Microsoft Windows, мова програмування C#, середовища Microsoft Visual Studio, Microsoft SQL Server Management Studio.

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) _проаналізувати існуючі проблеми безпеки даних та програмні засоби щодо їх вирішення, обґрунтувати обрані програмні додатки та способи розробки програмних застосунків, розробити програмне забезпечення, розробити

користувацький інтерфейс, зробити висновки за результатами проведеної роботи

5. Перелік ілюстративного матеріалу 1) Аналіз систем безпеки сучасних баз даних; 2) Архітектура СУБД; 3) Засоби розробки програмного продукту; 4) Взаємодія користувача із програмною системою.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання "11" березня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	23.03.20	
2.	Розробка архітектури та загальної структури системи	15.04.20	
3.	Розробка структур окремих підсистем	06.05.20	
4.	Програмна реалізація системи	27.05.20	
5.	Оформлення пояснювальної записки	03.06.20	
6.	Захист програмного продукту	10.06.20	
7.	Передзахист	10.06.20	
8.	Захист	17.06.20	

Студент _____ Поліно В.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Отрох С.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Мета роботи — дослідження систем та засобів захисту даних та їх застосування для безпечного управління даними в сховищах інформаційних ресурсів. Для створення інтерфейсу використано засоби візуального програмування Windows Forms. Для створення функціонала додатку використано засоби мови C#.

Розроблений програмний комплекс забезпечує керування агенту споживача електроенергії.

Записка містить 65 сторінок, 21 рисунків, 3 таблиці, 3 додатки і 11 посилань.

Ключові слова: бази даних, архітектура, засоби захисту даних, система безпечного керування даними, інформаційна безпека, системи управління базами даних, збережені процедури, цілісність метаданих.

ABSTRACT

The purpose of the work is studying data protection systems and their using for secure data management in information resource repositories. Windows Forms visual programming tools were used to create the interface. C # language tools were used to create the application's functionality.

The developed software package provides control of the agent of the consumer of the electric power.

The note contains 65 pages, 21 drawings, 3 tables, 3 applications and 11 links.

Keywords: databases, architecture, data protection tools, secure data management system, informational security, database management systems, stored procedures, metadata integrity.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

T-SQL – Transact Structured Query Language

ADO – ActiveX Data Objects

ERP – Enterprise Resource Planning

LINQ – Language–Integrated Query

XML – eXtensible Markup Language

CLR – Common Language Runtime

ВСТУП

На момент сьогодення використання баз даних є однією з характерних рис більшості сучасних інформаційних систем. В основі суті поняття “база даних” є те, без чого не можлива і навколо чого будуються інформаційні системи будь-яких підприємств. Можливо саме тому практиці використання баз даних та теорії їх створення приділяється досить велика увага протягом усього періоду функціонування інформаційних систем.

Протягом досить довгого періоду часу основним типом баз даних вважалися реляційні, які сьогодні вже сприймаються як класичні. Проте стрімкий розвиток інформаційних систем поставив завдання перед сучасними базами даних, вирішення яких потребує більш широких можливостей, ніж можуть запропонувати реляційні бази даних. Окрім класичних завдань, на сучасні бази даних покладаються завдання забезпечення багатомашинної обробки та зберігання великого об’єму інформації, інтеграції з мережею Інтернет, оперативного аналізу даних, розмежування доступу користувачів до інформації що зберігається, захисту інформації під час передачі її по мережі. Для забезпечення комфортної роботи з базами даних сьогодні існують різноманітні системи управління базами даних (СУБД).

СУБД по суті являє собою комплекс програм, що дозволяють створити базу даних (БД) і проводити маніпуляції над даними (оновлювати, вставляти, видаляти і вибирати). Система забезпечує безпеку, надійність зберігання і цілісність даних, а також надає кошти для адміністрування БД. Однак, наскільки всім добре відомо, “Жодна система не є ідеальною” і будь–яка СУБД не є виключенням.

В моєму проєкті за предметну область обрано систему безпеки бази даних, яка буде забезпечувати цілісність та недоторканість реєстру метаданих та обмежить його від несанкціонованих проникнень з боку небажаних

користувачів. Щоб забезпечити виконання поставленої цілі необхідно створити збережену процедуру, яка забезпечить виконання поставленого завдання.

В якості СУБД було обрано Microsoft SQL Server. Це пояснюється тим, що Microsoft SQL є однією із найпоширеніших і вдосконалених систем управління базами даних, що дозволить розроблений продукт без проблем впровадити до середовища використання, а також зробити її виконання автоматизованим.

Середовищами для створення обрано Microsoft Visual Studio. Для впровадження було використано функціонал середовища Microsoft SQL Management Studio. Також, для демонстрації роботи додатку буде створено тестову модель бази даних, що буде емулювати доступ до роботи з метаданими.

Окрім того, що розроблений програмний засіб може бути використаний будь-яким користувачем Microsoft SQL, він не вимагатиме від ПК нереальних системних характеристик, а для потенційного користувача не буде занадто тяжким завданням розібратися в правилах його використання.

1 АНАЛІЗ СИСТЕМ БЕЗПЕКИ СУЧАСНИХ БАЗ ДАНИХ

Персональні дані співробітників, фінансова інформація, внутрішня операційна інформація компанії, інформація про клієнтів і замовників, інтелектуальна власність, дослідження ринку, аналіз діяльності конкурентів, платіжна інформація – це відомості, якими найчастіше цікавляться кіберзлочинці, і майже постійно вони зберігаються в корпоративних базах даних.

Саме через значимість, важливість та цінність цієї інформації виникає необхідність забезпечити захист не тільки для елементів інфраструктури, а й для самих баз даних. Спробуємо розглянути і комплексно систематизувати питання безпеки різних систем управління базами даних (надалі СУБД) в світлі нових загроз, загальних тенденцій розвитку інформаційної безпеки і їх зростаючу роль, а також їх різноманітності.

1.1 Аналіз сучасної ситуації у сфері безпеки баз даних

Список основних вразливостей СУБД не зазнав істотних змін за останні роки.

Проаналізувавши засоби забезпечення безпеки, архітектуру БД, відомі уразливості і інциденти безпеки, можна виділити наступні причини виникнення такої ситуації:

- серйозна зацікавленість у проблемах безпеки тільки з боку великих виробників;
- нестача належної уваги до питань безпеки з боку програмістів баз даних, прикладних програмістів і адміністраторів;
- потреба в різних підходах до питань безпеки через різні масштаби і види збережених даних;

- використання різними СУБД специфічних мовних конструкцій для доступу до даних, організованих на основі однієї і тієї ж моделі;
- поява нових видів і моделей зберігання даних.

За рахунок простої неуваги або незнання адміністраторами систем баз даних питань безпеки багато уразливостей досі зберігають свою актуальність. Наприклад, прості SQL-ін'єкції широко експлуатуються сьогодні по відношенню до різних web-додатків, в яких не приділяється достатня увага до вхідних даних запитів.

Ці проблеми посилюються з появою і широким поширенням нереляційних СУБД, що оперують іншою моделлю даних, однак побудованих за принципами реляційних. Різноманіття сучасних NoSQL-рішень призводить до різноманітності застосовуваних моделей даних і розмиває межу поняття БД.

Наслідком цих проблем і відсутності єдиних методик є нинішня ситуація з безпекою NoSQL–систем. У більшості з них відсутні не тільки загальноприйняті механізми безпеки на кшталт шифрування, підтримки цілісності та аудиту даних, але й навіть розвинені засоби аутентифікації користувачів.

Застосування різних засобів забезпечення інформаційної безпеки є для організації таким собі компромісом у фінансовому плані: впровадження більш захищених продуктів і підбір більш кваліфікованого персоналу вимагають великих витрат. Окрім того, компоненти безпеки часто можуть негативно впливати на продуктивність СУБД.

1.2 Опис основних видів загроз даних БД

Атаки на БД і сховища найбільш небезпечними для великих і середніх організацій. В останні роки число витоків стрімко збільшується, причому не менше 30% порушень цілісності даних пов'язано саме із зовнішнім втручанням. Для аналізу загрози умовно можна застосувати наступну схему (рис. 1.1):

Рисунок 1.1 – Схема визначення загрози інформаційної безпеки даних

Такий підхід до аналізування загроз, при проектуванні систем захисту, допоможе більш точно виділити цілі розробки та дозволить підвищити її ефективність.

Наразі не існує єдиної загальноприйнятої класифікації загроз, хоча існує багато її варіантів. Умовно, загрози можна розподілити на випадкові і навмисні (рис. 1.2).

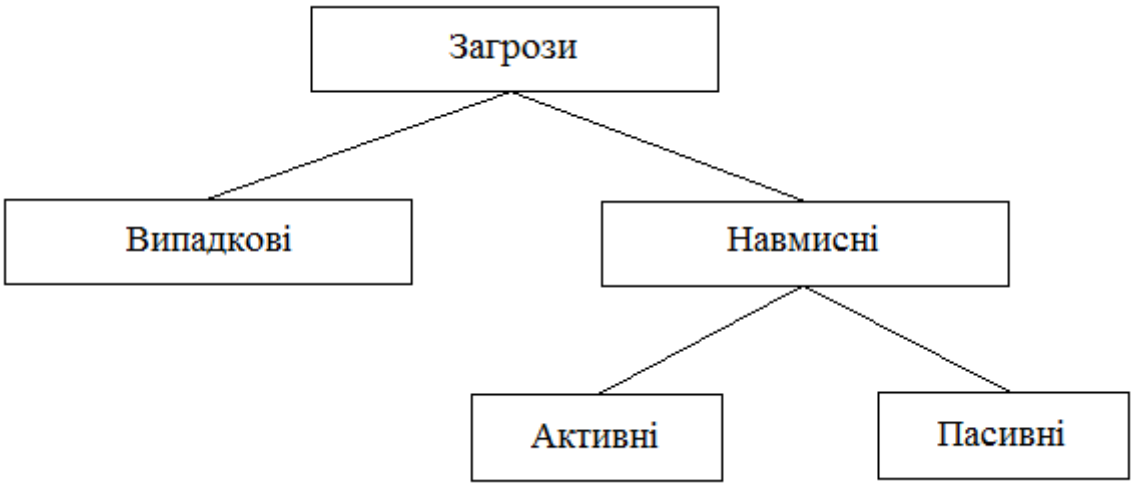

Рисунок 1.2 – Основний поділ загроз інформаційної безпеки

Джерелом випадкових можуть бути помилки в програмному забезпеченні, виходи з ладу апаратних засобів, неправильні дії користувачів або адміністрації мережі.

Навмисні загрози, на відміну від випадкових, прагнуть нанести шкоду користувачам і, в свою чергу, діляться на активні і пасивні.

Пасивні загрози, як правило, спрямовані на несанкціоноване використання інформаційних ресурсів мережі, не впливаючи при цьому на її функціонування. Наприклад, спроба отримання інформації, що циркулює в каналах передачі даної локальної обчислювальної мережі, шляхом підслуховування.

Активні загрози прагнуть порушити нормальне функціонування системи шляхом цілеспрямованого впливу на її апаратні, програмні і інформаційні ресурси. До активних загроз відносяться, наприклад, порушення або радіоелектронне заглушення ліній зв'язку локальної обчислювальної мережі, вивід з ладу пристроїв або її операційної системи, спотворення відомостей в користувацьких базах даних або системної інформації локальної обчислювальної мережі. Джерелами активних загроз можуть бути безпосередні дії зловмисників, програмні віруси.

Якщо роздивлятись класифікацію більш широко, виділяють наступний перелік класифікацій:

- за ціллю реалізації;
- за принципом дії на систему;
- за характером впливу на систему;
- за причиною появи помилки захисту;
- за способом дії атаки на об'єкт;
- за об'єктом атаки;
- за використовуваними засобами атаки;
- за станом об'єкту атаки.

Як правило, кіберзлочинців найчастіше цікавлять персональні дані співробітників, інформація про клієнтів та замовників, фінансова та платіжна

інформація, результати досліджень ринку, аналіз діяльності конкурентів та інші відомості, які практично завжди є в корпоративних базах даних.

З огляду на особливу значущість і цінність такої інформації, виникає необхідність в підвищенні безпеки як елементів інфраструктури, так і, власне, самих баз даних (БД).

1.3 Опис загальних принципів створення захищених БД

В основі понять про захищені БД неодмінно слід виділити основне, а саме поняття інформаційної безпеки.

Інформаційна безпека – захищеність інформації та інфраструктури, що її підтримує, від випадкових або навмисних дій природного або штучного характеру, які можуть завдати неприйнятного збитку суб'єктам інформаційних відносин, зокрема власникам і користувачам інформації та інфраструктури, що її підтримує.

Дане поняття є багатогранним і успіх у його вирішенні може принести тільки комплексний підхід, а саме золоте правило “Забезпечення існування трьох китів”. Дане порівняння є дуже доречним, оскільки інформаційна безпека тримається саме на трьох основних поняттях, тобто на доступності, цілісності і конфіденційності інформаційних ресурсів (рис. 1.3).

Рисунок 1.3 – Основні складові інформаційної безпеки

Під доступністю мається на увазі можливість за прийнятний час одержати необхідну інформаційну послугу.

Цілісність передбачає актуальність і несуперечність інформації, її захищеність від руйнування і несанкціонованої зміни.

Конфіденційність – це захист від несанкціонованого доступу до інформації.

Дотримання цього правила при розробці програмного забезпечення (ПЗ) дозволить максимально ефективно забезпечувати виконання майбутнім продуктом його функцій та завдань.

1.4 Постановка задачі

Проаналізувавши основні аспекти розробки, можемо поставити конкретне завдання для розроблюваного ПЗ.

За мету взято створення додаткового програмного забезпечення для СУБД Microsoft SQL Server, яке має надати можливість захисту цілісності та недоторканості каталогу метаданих від небажаних вторгнень з боку третіх осіб.

При розробленні програмного продукту необхідно створити тестову БД в SQL Server та заповнити їх необхідними полями (атрибутами) для майбутніх тестувань та демонстрацій.

Функціонал створеного забезпечення зможе використовувати деякі можливості вже існуючих процедур, а також інших існуючих засобів і можливостей SQL Server, такі як сертифікати, запити, представлення тощо.

2 АРХІТЕКТУРНІ ОСОБЛИВОСТІ СИСТЕМ УПРАВЛІННЯ БАЗАМИ ДАНИХ ПРИ РОЗРОБЦІ БЕЗПЕКОВИХ СИСТЕМ

У зв'язку з тим, що поняття бази даних спочатку передбачало можливість вирішення багатьох завдань кількома користувачами, найважливішою характеристикою сучасних СУБД є наявність багатокористувацької технології роботи. Різна реалізація таких технологій в різний час була пов'язана як з основними властивостями обчислювальної техніки, так і з розвитком програмного забезпечення і систем безпеки. Далі буде наведена коротка характеристика існуючих архітектур та їх основні властивості.

2.1 Централізована архітектура

При використанні цієї технології база даних, СУБД і прикладна програма (додаток) розташовуються на одному комп'ютері (персональному комп'ютері або мейнфреймі) (рис. 2.1).

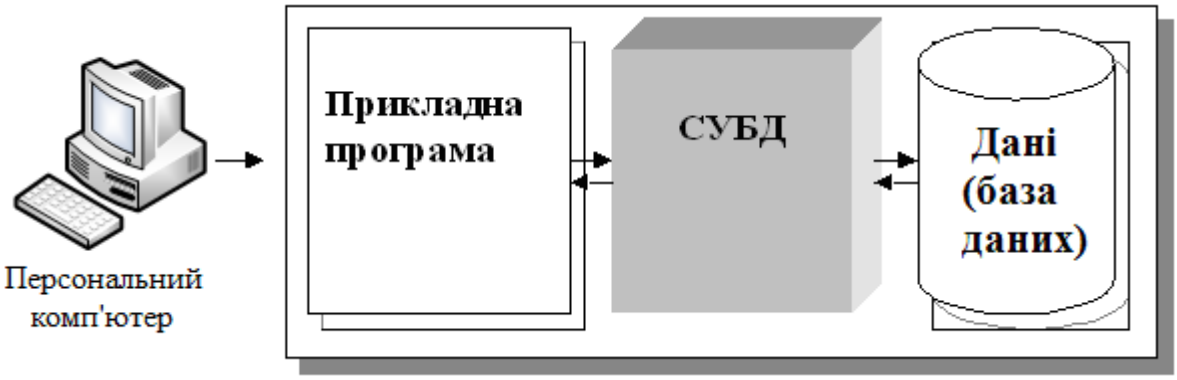

Рисунок 2.1 – Централізована архітектура

Подібна архітектура використовувалася в перших версіях СУБД DB2, Oracle, Ingres [1].

Для такого способу організації нема необхідності у підтримці мережі і все зводиться до автономної роботи. Робота побудована таким чином:

- База даних у вигляді набору файлів знаходиться на жорсткому диску комп'ютера.
- На тому ж комп'ютері встановлено СУБД і додаток для роботи з БД.
- Користувач запускає додаток. Використовуючи надається додатком призначений для користувача інтерфейс, він ініціює звернення до БД на вибірку / оновлення інформації.
- Всі звернення до БД йдуть через СУБД, яка інкапсулює в собі всі відомості про фізичну структуру БД.
- СУБД ініціює звернення до даних, забезпечуючи виконання запитів користувача (здійснюючи необхідні операції над даними).
- Результат СУБД повертає в додаток.
- Додаток, використовуючи призначений для користувача інтерфейс, відображає результат виконання запитів.

Розрахована на багато користувачів технологія роботи забезпечувалася або режимом мультипрограмування (одночасно могли працювати процесор і зовнішні пристрої - наприклад, поки в прикладній програмі одного користувача йшло зчитування даних з зовнішньої пам'яті, програма іншого користувача оброблялася процесором), або режимом поділу часу (користувачам по черзі виділялися кванти часу на виконання їх програм). Така технологія була поширена в період "панування" великих ЕОМ. Основним недоліком цієї моделі є різке зниження продуктивності при збільшенні числа користувачів.

2.2 Архітектура “файл–сервер”

Збільшення складності завдань, поява персональних комп'ютерів і локальних обчислювальних мереж з'явилися передумовами появи нової архітектури файл-сервер. Ця архітектура баз даних з мережевим доступом передбачає призначення одного з комп'ютерів мережі в якості виділеного

сервера, на якому будуть зберігатися файли бази даних [1]. Відповідно до запитів користувачів файли з файл-сервера передаються на робочі станції користувачів, де і здійснюється основна частина обробки даних. Центральний сервер виконує в основному тільки роль сховища файлів, не беручи участь в обробці самих даних (рис. 2.2).

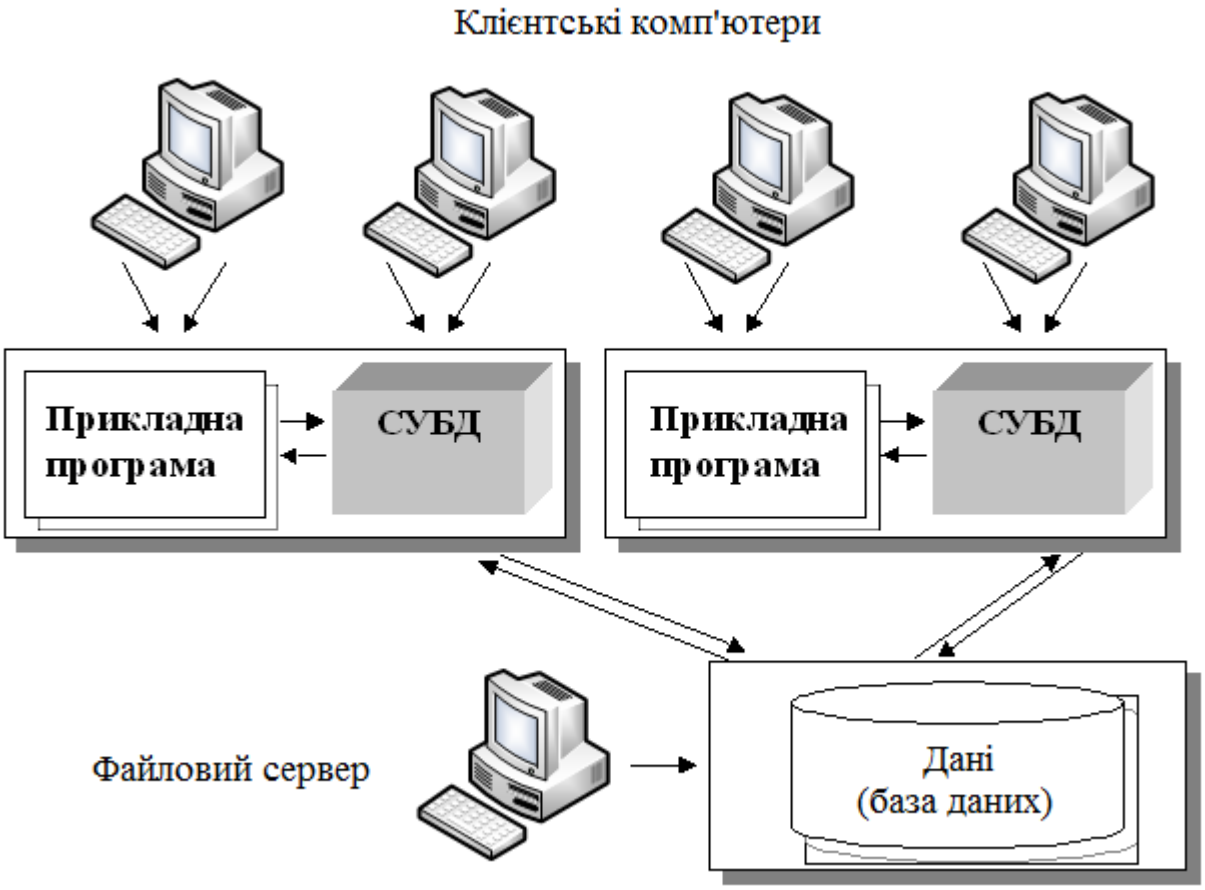

Рисунок 2.2 – Архітектура “файл–сервер”

Робота побудована таким чином:

- База даних у вигляді набору файлів знаходиться на жорсткому диску спеціально виділеного комп'ютера (файлового сервера).
- Існує локальна мережа, що складається з клієнтських комп'ютерів, на кожному з яких встановлені СУБД і додаток для роботи з БД.
- На кожному з клієнтських комп'ютерів користувачі мають можливість запустити додаток. Використовуючи надається додатком призначений

для користувача інтерфейс, він ініціює звернення до БД на вибірку / оновлення інформації.

- Всі звернення до БД йдуть через СУБД, яка інкапсулює в собі всі відомості про фізичну структуру БД, розташованої на файловому сервері.
- СУБД ініціює звернення до даних, що знаходяться на файловому сервері, в результаті яких частина файлів БД копіюється на клієнтський комп'ютер і обробляється, що забезпечує виконання запитів користувача (здійснюються необхідні операції над даними).
- При необхідності (в разі зміни даних) дані відправляються назад на файловий сервер з метою оновлення БД.
- Результат СУБД повертає в додаток.
- Додаток, використовуючи призначений для користувача інтерфейс, відображає результат виконання запитів.

В рамках архітектури "файл-сервер" були виконані перші версії популярних так званих настільних СУБД, таких, як dBase і Microsoft Access.

Далі вказуються основні недоліки даної архітектури:

- При одночасному зверненні безлічі користувачів до одних і тих же даних продуктивність роботи різко падає, тому що необхідно дочекатися поки користувач, що працює з даними, завершить свою роботу. В іншому випадку можливе затирання виправлень, зроблених одними користувачами, змінами інших користувачів.
- Весь тягар обчислювального навантаження при доступі до БД лягає на додаток клієнта, так як при видачі запиту на вибірку інформації з таблиці вся таблиця БД копіюється на клієнтську машину і вибірка здійснюється на клієнті. Таким чином, не оптимально витрачаються ресурси клієнтського комп'ютера і мережі. В результаті зростає мережевий трафік і збільшуються вимоги до апаратних потужностей для користувача комп'ютера.

- Як правило, використовується навігаційний підхід, орієнтований на роботу з окремими записами.
- В БД на файл-сервері набагато простіше вносити зміни в окремі таблиці, минаючи програми, безпосередньо з інструментальних засобів (наприклад, з утиліти Database Desktop фірми Borland для файлів Paradox і dBase); подібна можливість полегшується тією обставиною, що фактично у таких СУБД база даних - поняття більш логічне, ніж фізичне, оскільки під БД розуміється набір окремих таблиць, що співіснують в окремому каталозі на диску. Все це дозволяє говорити про низький рівень безпеки - як з точки зору розкрадання та завдання шкоди, так і з точки зору внесення помилкових змін.
- Недостатньо розвинений апарат транзакцій служить потенційним джерелом помилок в плані порушення смислової і посилальної цілісності інформації при одночасному внесенні змін в одну і ту ж запис.

2.3 Архітектура “клієнт–сервер”

Використання технології "клієнт – сервер" передбачає наявність певної кількості комп'ютерів, об'єднаних в мережу, один з яких виконує особливі керуючі функції (є сервером мережі).

Так, архітектура "клієнт - сервер" розділяє функції програми користувача (званого клієнтом) і сервера. Додаток-клієнт формує запит до сервера, на якому розташована БД, на структурному мовою запитів SQL (Structured Query Language), що є промисловим стандартом у світі реляційних БД. Віддалений сервер приймає запит і переадресує його SQL-серверу БД. SQL–сервер – спеціальна програма, що керує віддаленою базою даних. SQL–сервер забезпечує інтерпретацію запиту, його виконання в базі даних, формування результату виконання запиту та видачу його додатку-клієнту. При цьому ресурси

клієнтського комп'ютера не беруть участь у фізичному виконанні запиту; клієнтський комп'ютер лише відсилає запит до серверної БД і отримує результат, після чого інтерпретує його необхідним чином і являє користувачу. Так як клієнтського додатку надсилається результат виконання запиту, по мережі "подорожують" тільки ті дані, які необхідні клієнту. В результаті знижується навантаження на мережу. Оскільки виконання запиту відбувається там же, де зберігаються дані (на сервері), немає необхідності в пересиланні великих пакетів даних. Крім того, SQL–сервер, якщо це можливо, оптимізує отриманий запит таким чином, щоб він був виконаний в мінімальний час з найменшими накладними витратами [1]. Архітектура системи представлена на рис. 2.3.

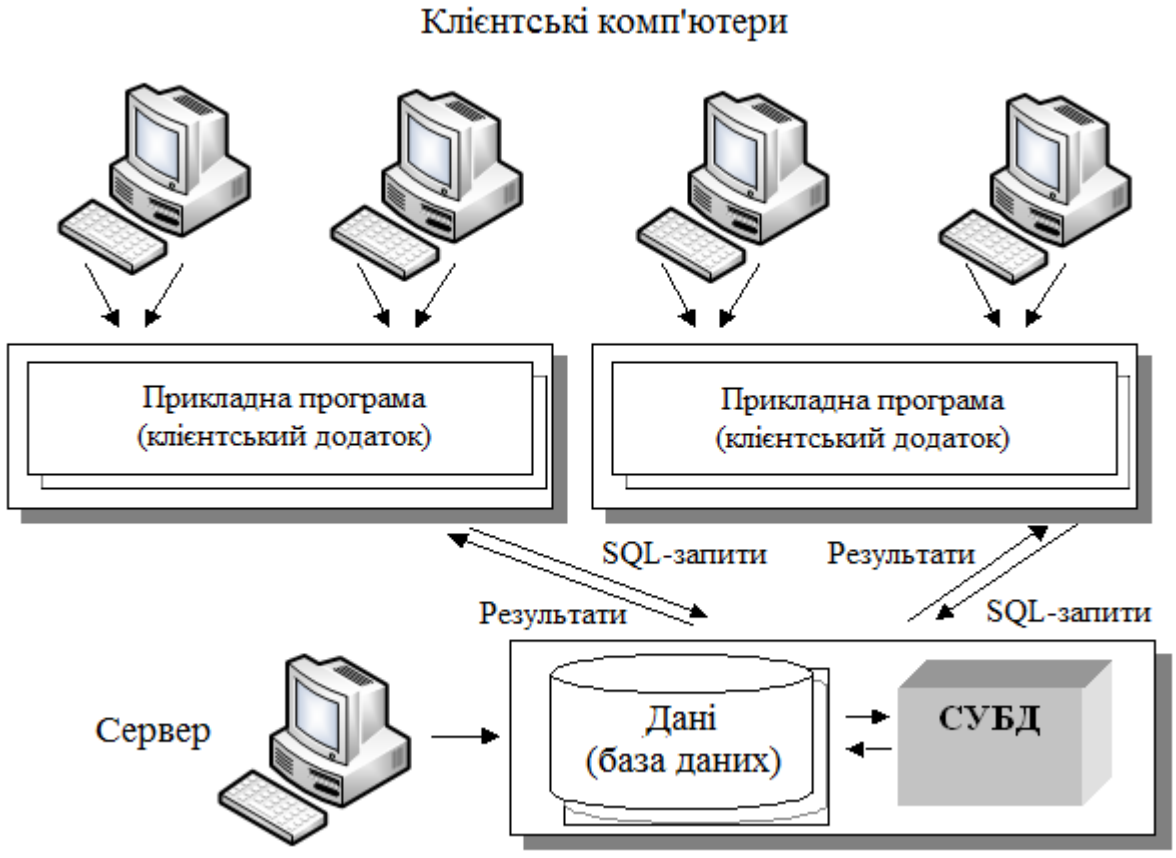

Рисунок 2.3 – Архітектура “клієнт–сервер”

Отже, в результаті робота побудована наступним чином:

- База даних у вигляді набору файлів знаходиться на жорсткому диску спеціально виділеного комп'ютера (сервера мережі).
- СУБД розташовується також на сервері мережі.

- Існує локальна мережа, що складається з клієнтських комп'ютерів, на кожному з яких встановлено клієнтське додаток для роботи з БД.
- На кожному з клієнтських комп'ютерів користувачі мають можливість запустити додаток. Використовуючи надається додатком призначений для користувача інтерфейс, він ініціює звернення до СУБД, розташованої на сервері, на вибірку / оновлення інформації. Для спілкування використовується спеціальна мова запитів SQL, тобто по мережі від клієнта до сервера передається лише текст запиту.
- СУБД інкапсулює всередині себе всі відомості про фізичну структуру БД, розташованої на сервері.
- СУБД ініціює звернення до даних, що знаходяться на сервері, в результаті яких на сервері здійснюється вся обробка даних і лише результат виконання запиту копіюється на клієнтський комп'ютер. Таким чином СУБД повертає результат в додаток.
- Додаток, використовуючи призначений для користувача інтерфейс, відображає результат виконання запитів.

В архітектурі "клієнт–сервер" працюють так звані "промислові" СУБД. Промисловими вони називаються через те, що саме СУБД цього класу можуть забезпечити роботу інформаційних систем масштабу середнього і великого підприємства, організації, банку. До розряду промислових СУБД належать MS SQL Server, Oracle, Gupta, Informix, Sybase, DB2, InterBase і ряд інших [1].

Розглянемо основні переваги даної архітектури в порівнянні з архітектурою "файл-сервер":

- Істотно зменшується мережевий трафік.
- Зменшується складність клієнтських додатків (велика частина навантаження лягає на серверну частину), а, отже, знижуються вимоги до апаратних потужностей клієнтських комп'ютерів.

- Наявність спеціального програмного засобу – SQL–сервера – призводить до того, що значна частина проектних і програмістів завдань стає вже вирішеною.
- Істотно підвищується цілісність і безпека БД.

До числа недоліків можна віднести більш високі фінансові витрати на апаратне і програмне забезпечення, а також те, що велика кількість клієнтських комп'ютерів, розташованих в різних місцях, викликає певні труднощі зі своєчасним оновленням клієнтських додатків на всіх комп'ютерах-клієнтах. Проте, архітектура "клієнт - сервер" добре зарекомендувала себе на практиці, в даний момент існує і функціонує велика кількість БД, побудованих відповідно до даної архітектурою.

2.4 Багатоланкова архітектура

Триланкова (в деяких випадках багатоланкова) архітектура (N–tier або multi-tier). є подальше вдосконалення технології "клієнт – сервер". Розглянувши архітектуру "клієнт – сервер", можна зробити висновок, що вона є 2-звенной: перша ланка – клієнтську програму, друга ланка – сервер БД + сама БД. У триланкової архітектурі вся бізнес-логіка (ділова логіка), яка раніше входила до клієнтські програми, виділяється в окрему ланку, зване сервером додатків. При цьому клієнтським застосуванням залишається лише користувальницький інтерфейс. Так, в якості клієнтського додатку в описаному вище прикладі виступає Web-браузер.

Що поліпшується при використанні триланкової архітектури? Тепер при зміні бізнес-логіки більш немає необхідності змінювати клієнтські програми і оновлювати їх у всіх користувачів. Крім того, максимально знижуються вимоги до апаратури користувачів.

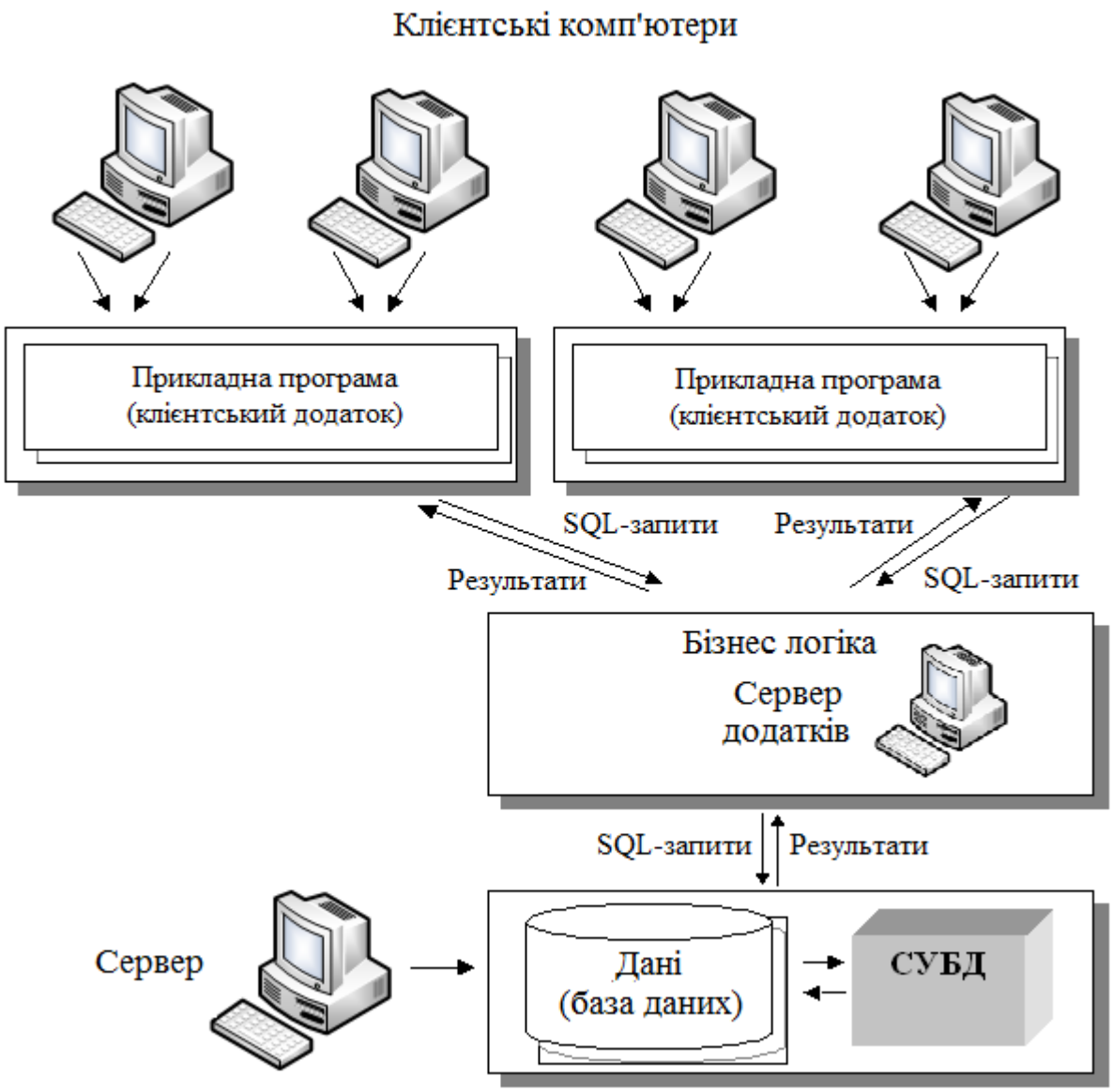

Рисунок 2.4 – Багатоланкова архітектура “клієнт–сервер”

Отже, в результаті робота побудована наступним чином:

- База даних у вигляді набору файлів знаходиться на жорсткому диску спеціально виділеного комп'ютера (сервера мережі).
- СУБД розташовується також на сервері мережі.
- Існує спеціально виділений сервер додатків, на якому розташовується програмне забезпечення (ПО) ділового аналізу (бізнес-логіка).
- Існує безліч клієнтських комп'ютерів, на кожному з яких встановлений так званий "тонкий клієнт" - клієнтську програму, що реалізує інтерфейс користувача.

- На кожному з клієнтських комп'ютерів користувачі мають можливість запустити додаток - тонкий клієнт. Використовуючи надається додатком призначений для користувача інтерфейс, він ініціює звернення до ПО ділового аналізу, розташованому на сервері додатків.
- Сервер додатків аналізує вимоги користувача і формує запити до БД. Для спілкування використовується спеціальна мова запитів SQL, тобто по мережі від сервера додатків до сервера БД передається лише текст запиту.
- СУБД інкапсулює всередині себе всі відомості про фізичну структуру БД, розташованої на сервері.
- СУБД ініціює звернення до даних, що знаходяться на сервері, в результаті яких результат виконання запиту копіюється на сервер додатків.
- Сервер додатків повертає результат в клієнтську програму (користувачеві).
- Додаток, використовуючи призначений для користувача інтерфейс, відображає результат виконання запитів.

2.5 Огляд ринку СУБД

Багато авторів класифікують СУБД на дві великі категорії: так звані “настільні” і “серверні”.

Настільні СУБД використовуються для порівняно невеликих завдань (невеликий обсяг оброблюваних даних, мала кількість користувачів). З огляду на це, зазначені СУБД мають відносно спрощену архітектуру, зокрема, функціонують в режимі файл-сервер, підтримують не всі можливі функції СУБД (наприклад, не ведеться журнал транзакцій, відсутня можливість автоматичного відновлення бази даних після збоїв і т.д.). Проте, такі системи мають досить велику область застосування. Перш за все, це державні (муніципальні) установи, сфера освіти, сфера обслуговування, малий і середній бізнес. Специфіка

виникаючих там завдань полягає в тому, що обсяги даних не є катастрофічно великими, частота оновлень не буває занадто високою, організація територіально зазвичай розташована в одній невеликій будівлі, кількість користувачів коливається від одного до 10-15 чоловік. У подібних умовах використання настільних СУБД для управління інформаційними системами є цілком виправданим, і вони з успіхом застосовуються.

В останні роки дуже широке поширення набула система управління базами даних Microsoft Access, яка входить в цілий ряд версій пакета Microsoft Office (фірма Microsoft).

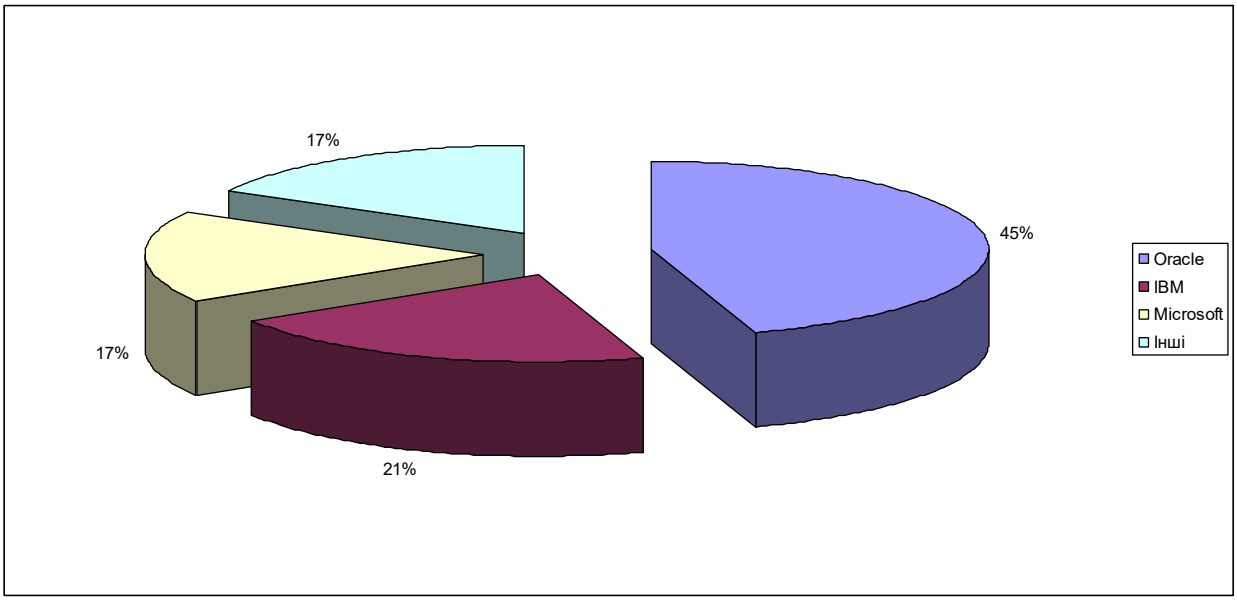

Рисунок 2.5 – Співвідношення використання систем зберігання даних у світі

Найбільш поширеними клієнт-серверними системами тут відповідно є системи Oracle (розробник компанія Oracle), MS SQL Server (розробник компанія Microsoft), Informix Dynamic Server (компанія IBM).

3 АНАЛІЗ ТА ОБҐРУНТУВАННЯ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Актуальність розробки систем безпеки баз даних

Наразі притаманним для майже всіх великих виробників СУБД є обмеження на етапі розвитку концепції конфіденційності, доступності та цілісності даних, а свої дії вони спрямовують, в основному, на подолання вже існуючих вразливостей, реалізацію основних моделей доступу і розгляд специфічних для конкретної СУБД питань. Використання такого підходу забезпечує вирішення конкретних завдань, але не сприяє появі загальної концепції безпеки у такому класі ПЗ, як СУБД. Це значно ускладнює завдання забезпечення безпеки у сховищах даних підприємства.

Можна виділити наступні архітектурні підходи [2]:

- повний доступ всіх користувачів до сервера БД;
- поділ користувачів засобами СУБД на довірених і частково довірених;
- введення системи аудиту (логів дій користувачів) засобами СУБД;
- введення шифрування даних;
- винос засобів аутентифікації за межі СУБД в операційні системи і проміжне ПЗ;
- відмова від повністю довіреного адміністратора даних.

Введення засобів захисту як реакції на загрози не забезпечує захист від нових способів атак і формує досить розрізнене уявлення про саму проблематику забезпечення безпеки.

Сховища даних включають в себе два компоненти:

- збережені дані (власне БД);
- програми управління (СУБД).

Забезпечення безпеки інформації, що зберігається, зокрема, неможливе без забезпечення безпечного управління даними. Виходячи з цього, всі вразливі

місця і питання безпеки СУБД діляться на дві категорії: залежні від даних і незалежні від даних.

Вразливості, що не залежать від даних, характерні також для всіх інших видів ПЗ. Їх причиною, наприклад, може стати наявність невикористовуваних функцій, несвоєчасне оновлення ПЗ або нестача кваліфікації адміністраторів ПЗ.

Більшість аспектів з питань безпеки СУБД є саме залежними від даних. У той же час багато вразливостей є побічно залежними від даних. Наприклад, більшість СУБД підтримують запити до даних за допомогою деякої мови запитів, що містить доступні користувачеві набори функцій (які, в свою чергу, теж можна вважати операторами мови запитів) або довільні функції на деякій мові програмування.

Архітектура вживаних мов, принаймні, що стосується спеціалізованих мов і наборів функцій, безпосередньо може бути пов'язана з моделлю даних, яка застосовується для зберігання інформації. Таким чином, модель визначає особливості мови, і наявність в ній тих чи інших вразливостей. Причому такі уразливості, як ін'єкція, виконуються, в залежності від синтаксису мови, по-різному (java-ін'єкція, sql-ін'єкція).

3.2 Система безпеки MS SQL

Такі загальні загрози безпеки, як крадіжка даних або вандалізм, існують незалежно від використовуваної версії SQL Server. Цілісність даних також слід розглядати як проблему безпеки. При відсутності захисту даних вони можуть стати марними, якщо дозволено нерегламентоване управління даними і в дані випадково або навмисно вносяться невірні значення або вони повністю видаляються. Крім того, часто існують законодавчі вимоги, яких необхідно дотримуватися, наприклад, що стосуються правильного зберігання конфіденційної інформації. Зберігання деяких особистих даних повністю заборонено в залежності від законодавства, яке застосовується в конкретній юрисдикції.

У кожній версії SQL Server є свої засоби безпеки, як і в кожної версії Windows, при цьому можливості більш пізніх версій ширше, ніж можливості більш ранніх. Важливо розуміти, що самі по собі засоби безпеки не можуть гарантувати захист програми бази даних. Кожна програма бази даних має унікальні вимоги, середовище виконання, модель розгортання, фізичне розташування і кількість користувачів. Деяким програмам, які працюють локально, необхідний мінімальний захист, тоді як іншим локальним додаткам або додаткам, розгорнутим через Інтернет, можуть знадобитись суворі заходи безпеки разом з постійним моніторингом та контролем.

Крім того, в сам SQL Server 2005 інтегровані багато нових або поліпшених можливостей, що вимагають підвищеної захищеності. Спеціалізовані можливості потребують спеціалізованих параметрів безпеки. Деякі з цих нових або поліпшених можливостей, такі як інтегрована підтримка кінцевих точок HTTP, вимагають, щоб SQL Server підтримував додаткові функції забезпечення безпеки. Інші нові можливості не потребували додаткових заходів захисту, але ці заходи все ж були реалізовані, щоб SQL Server став ще більш захищеним.

Далі наведено реалізовані в SQL Server функції захисту, які потрібно знати:

- можливість застосування політик безпеки операційної системи;
- нова ієрархічна модель дозволів;
- поділ схеми і власника;
- декларативний контекст виконання;
- управління безпекою доступу до метаданих;
- більш деталізована система дозволів.

Для розуміння суті змін і їх використання при розробці додатків розглянемо основи системи безпеки сервера. Основи системи безпеки MS SQL 2005 складають три поняття: суб'єкти системи, або учасники (Principals); об'єкти системи (Securables) і система дозволів (Permissions). У системах забезпечення безпеки розглядаються три базових етапи для отримання доступу суб'єкта до захищених об’єктів:

- можливість застосування політик безпеки операційної системи;
- ідентифікація виконується присвоєнням суб'єкту якого! або формального ідентифікатора, наприклад імені облікового запису (login) і пароля (password), або видачею йому смарткарти, що містить ключ шифрування, або виконанням аналізу відбитків пальців, або іншими способами;
- аутентифікація процес встановлення взаємооднозначної відповідності між суб'єктом і вводяться їм ідентифікаційними даними;
- авторизація процедура забезпечення суб'єкту доступу до елементів системи. Сам доступ регулюється дозволами.

Вимоги безпеки для програми бази даних SQL Server слід розглядати під час розробки, а не згодом. Оцінка загроз ще на стадії розробки дозволяє скоротити потенційний збиток в разі виявлення уразливості.

3.3 Суб’єкти та об’єкти системи безпеки (Security Principals & Securables)

Суб'єкт – це сутність, що пройшла процедуру аутентифікації. Після проходження вона може отримати доступ до ресурсів сервера на підставі своїх дозволів [3].

Суб'єкти можуть бути індивідуальними і груповими.

Створюються суб'єкти на трьох рівнях – на рівні операційної системи (Windows), SQL Server і бази даних (табл. 3.1).

Табл. 3.1. Структура суб'єктів системи безпеки

Операційна система	**Сервер**	**База Даних**
Обліковий запис користувача	Обліковий запис підключення	Обліковий запис користувача
Обліковий запис локальної групи	Обліковий запис підключення	Обліковий запис користувача

Облікові записи доменних груп	Обліковий запис підключення	Обліковий запис користувача

Створюючи різні суб'єкти, ви можете планувати систему безпеки і гнучко нею керувати.

Облікові записи можуть об'єднуватися в ролі.

Об'єкти системи безпеки – це елементи системи, що захищаються системою авторизації через призначені дозволи [3].

Система об'єктів ієрархічна, тобто об'єкти вкладені одна в одну і включає в себе сервер, бази даних, схеми (табл. 3.2).

Табл. 3.2. Об'єкти системи безпеки

Сервер	**База Даних**	**Схема**
Облікові записи підключення (login)	Облікові записи користувачів (user)	Таблиці (tables)
HTTP endpoint	Ролі (roles)	Перегляди (views)
Сертифікати (certificates)	Ролі додатки (application roles)	Функції (functions)
Повідомлення про події (event notifications)	Складання (assemblies)	Процедури (storage procedure)
Бази даних (databases)	Типи повідомлень (message types)	Черги (queues)
	Контракти Service Broker (service contracts)	Типи даних (types)
	Служби (services)	Правила (rules)
	Каталоги системи (full! Text catalogs)	Значення за замовчуванням повнотекстового пошуку (defaults)

	DDL-події (DDL event)	Синоніми (synonyms)
	Схема (schema)	Агрегати (aggregates)

По суті, об'єкти – це ресурси SQL Server, захищені дозволами.

Об'єкти мають різну область визначення і дозволяють вам гнучко планувати свою систему безпеки.

3.4 Система дозволів

У суб'єкта системи є тільки один шлях отримання доступу до об'єктів – мати призначені безпосереднього або опосередковано дозволу.

При безпосередньому управлінні дозволами вони призначаються явно суб'єкту, а при опосередкованому дозволу призначаються через членство в групах, ролях або успадковуються від об'єктів, що лежать вище по ланцюжку ієрархії.

За своєю суттю дозволи – це набір правил, який забезпечує необхідний рівень доступу до об'єктів. Дозволи можуть бути видані (grant), заборонені (deny) і вилучені (revoke) [3].

У SQL Server введена можливість керувати дозволами на сервері (для серверних об'єктів). У попередній версії така можливість була реалізована тільки за рахунок фіксованих ролей сервера.

Види дозволів залежать від ряду факторів, у тому числі від області (сервер, база, схема), від типу оператора, на який призначаються дозволу. У таблиці 3.3 наведені деякі види дозволів, які призначаються на різних рівнях сервера.

Табл. 3.3. Види дозволів

Сервер	**База даних**	**Схема**
CONNECT_SQL	CREATE TABLE	SELECT
CREATE LOGIN	ALTER ANY USER	ALTER
ALTER ANY LOGIN	CONTROL	TAKE OWNERSHIP

CONTROL SERVER		

Крім того, підтримується спадкування дозволів від об'єктів вниз по ієрархії. Наприклад, встановлений для будь-кого дозвіл CONTROL на базу даних успадковується всіма об'єктами бази аж до самого нижнього по ієрархії.

Управління дозволами в базі виконується на двох рівнях:

- управління дозволами на виконання певних завдань всередині бази;
- управління дозволами на об'єктах безпеки, що знаходяться в базі.

3.5 Доступ до метаданих

Кожна СУБД зберігає метадані про всі сутності бази даних. Так в SQL Server за допомогою інструкції CREATE можна створити 52 сутності [4]. Наприклад:

- CREATE ASSEMBLY;
- CREATE INDEX;
- CREATE TABLE;
- CREATE FUNCTION;
- CREATE DATABASE;

У різних СУБД застосовуються різні назви для метаданих - системний каталог, словник даних тощо. Однак загальною властивістю всіх сучасних реляційних СУБД є те, що каталог чи словник сам складається з таблиць, а якщо точніше - системних таблиць. В результаті користувач має змогу звертатися до метаданих так само, як і до прикладних даних, використовуючи інструкцію SELECT. Зміни ж в каталозі / словнику виробляються автоматично при виконанні відповідних інструкцій користувачем, що змінюють стан об'єктів бази даних. Системні таблиці не повинні змінюватися безпосередньо жодним користувачем. Microsoft SQL Server надає наступні колекції системних представлень, що містять метадані:

- представлення інформаційної схеми;
- представлення каталогу;
- представлення сумісності;
- представлення реплікації;
- динамічні адміністративні представлення і функції;
- подання додатка рівня даних (DAC).

Представлення інформаційної схеми визначаються в особливій схемі з ім'ям “INFORMATION_SCHEMA”. Ця схема міститься в будь-якій базі даних і складається з 20 представлень [4].

Наприклад:

- TABLES;
- VIEWS.

Представлення каталогів баз даних і файлів містить 6 представлень [4].

Наприклад:

- sys.database_mirroring;
- sys.database_recovery_status;
- sys.databases.

Також представлення MS SQL Server містять:

- Метадані серверного рівня (інформація про суб'єктів безпеки сервера, облікові записи сервера, дозволи, членство в ролях);
- Метадані рівня бази даних (інформація про суб'єктів безпеки бази даних, дозволи, членство в ролях);
- Метадані про шифрування (інформація про сертифікати, симетричні та асиметричні ключі).

Так само тут містяться:

- Системні збережені процедури;
- Функції метаданих;
- Функції конфігурації;
- Функції безпеки;

- Системні функції;
- DBCC команда.

3.6 Існуючі системні процедури в MS SQL Server

Системні збережені процедури існують в будь-якій більш-менш відомій СУБД. Не дивлячись на деякі спільні концепції, через різні підходи до їх розробки, вони суттєво відрізняються.

Процедура в SQL Server – це група з однієї або декількох інструкцій Transact-SQL або посилання на метод середовища CLR Microsoft .NET Framework. Процедури є аналогічними по конструкції до аналогів на інших мовах програмування, оскільки забезпечують наступне:

- обробка вхідних параметрів і повернення викликаного у програмі значення у вигляді вихідних параметрів;
- виконання операцій в базі даних, включаючи виклик інших процедур за допомогою вкладених програмних інструкцій;
- повернення значення стану викликаної програми, тим самим забезпечення передачі відомості про успішне або неуспішне завершення (і причини останнього).

У наступному списку описуються переваги використання даних процедур:

- зниження мережевого трафіку між клієнтами і сервером;
- висока надійність;
- повторне використання коду;
- легше обслуговування;
- підвищення продуктивності.

Збережені процедури мають вхідні параметри і повертають результати. На відміну від тригерів, які належать певній таблиці або поданням, збережені процедури належать базі даних в цілому. Вони можуть бути викликані будь-яким процесом, що використовують базу даних, за умови, що у цього процесу є достатні права доступу.

Збережені процедури використовуються для багатьох цілей. Хоча адміністратори баз даних використовують їх для виконання рутинних завдань адміністрування, головною областю їх застосування є в основному додатки баз даних.

3.7 Користувацькі збережені процедури в MS SQL Server

Користувацькі збережені процедури, як і системні, є повноцінними об’єктами бази даних. Політика компанії розробника у питанні впровадження користувацьких процедур є вкрай позитивною, адже розробник не тільки надає можливість створення додаткового забезпечення, а й заохочує даний процес.

При створенні збереженої процедури дозволяється визначити необов'язковий список параметрів. Таким чином, процедура буде приймати відповідні аргументи при кожному її виклику.

Створення збереженої процедури передбачає вирішення наступних завдань [5]:

- Визначення типу створюваної процедури: тимчасова або призначена для користувача. Крім цього, можна створити свою власну системну збережену процедуру, призначивши їй ім’я з префіксом sp_ і помістивши її в системну базу даних. Така процедура буде доступна в контексті будь-якої бази даних локального сервера;
- Планування прав доступу. При створенні збереженої процедури слід враховувати, що вона буде мати ті ж права доступу до об'єктів бази даних, що і створив її користувач;
- Визначення параметрів процедури. Подібно процедурам, які входять до складу більшості мов програмування, збережені процедури можуть мати вхідними та вихідними параметрами;
- Розробка коду збереженої процедури. Код процедури може містити послідовність будь-яких команд SQL, включаючи виклик інших процедур.

SQL Server підтримує загальномовне середовище виконання CLR (Common Language Runtime), яке дозволяє розробляти різні об'єкти баз даних (збережені процедури, тригери, визначені користувачем статистичні функції, призначені для користувача типи даних), застосовуючи мови C # і Visual Basic. Середовище CLR також дозволяє виконувати ці об'єкти, використовуючи систему загального середовища виконання.

Середовище CLR дозволяється і забороняється допомогою опції clr_enabled системної процедури sp_configure, яка запускається на виконання інструкцією RECONFIGURE. Нижче показано, як можна за допомогою системної процедури sp_configure дозволити використання середовища CLR (рис.4.1.):

```SQL
USE SampleDb;

EXEC sp_configure 'clr_enabled',1
RECONFIGURE
```

Рисунок 3.1 – Надання дозволу на використання середовища CLR

Для створення, компіляції та збереження процедури за допомогою середовища CLR виконується наступна послідовність кроків в зазначеному порядку:

1. Створити процедуру, що зберігається на мові C #, а потім скомпілювати її, використовуючи відповідний компілятор;
2. Використовуючи інструкцію CREATE ASSEMBLY, створити відповідний виконуваний файл;
3. Зберегти процедуру у вигляді об'єкта сервера, використовуючи інструкцію CREATE PROCEDURE;
4. Виконати процедуру, використовуючи інструкцію EXECUTE.

Наведені вище кроки дуже просто та чітко описує схема на рис. 3.2.

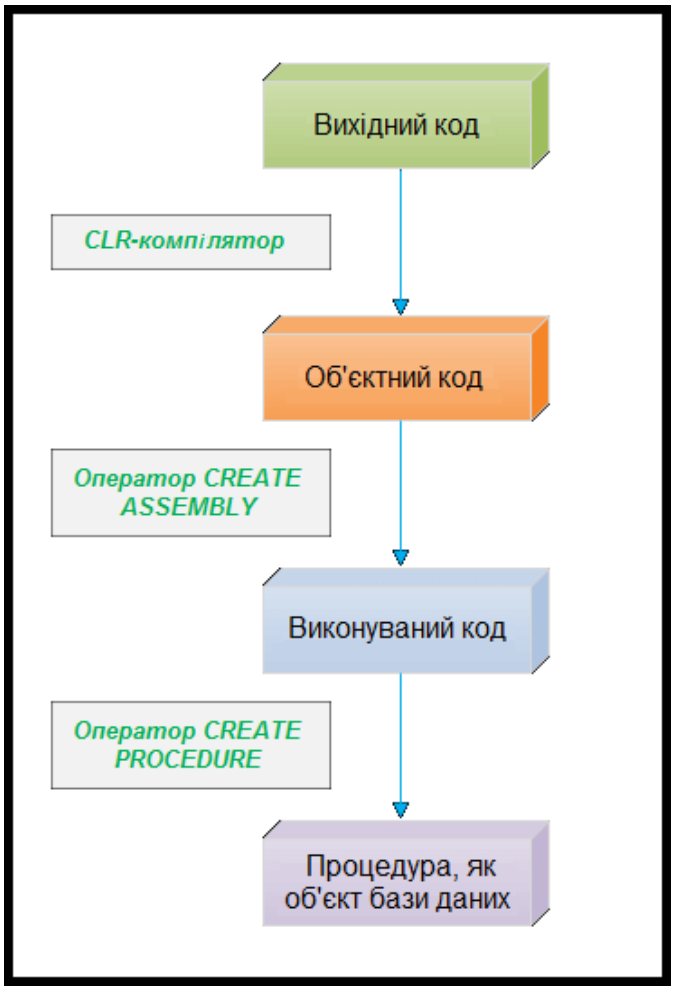

Рисунок 3.2 – Схема створення збереженої процедури

Як видно з наведеної вище схеми, процедура, перед збереженням у вигляді об'єкта в базі даних, попередньо компілюється.

В нашому випадку, основною ціллю, що буде покладена на збережену процедуру буде управління авторизацією доступу до роботи з метаданими.

4 ЗАСОБИ РОЗРОБКИ

4.1 Microsoft SQL Server

Microsoft SQL Server – система управління базами даних (СУБД), розроблена корпорацією Microsoft. Основна використовувана мова запитів – Transact-SQL, створена спільно Microsoft та Sybase [6]. Це повноцінна клієнт-серверна архітектура в якій по мережі не передаються самі таблиці представлення бази даних, а повертається результат певного запиту. Усі підрахунки відбуваються на сервері баз даних замість клієнтських комп'ютерів. Це значно збільшує швидкодію і це особливо важливо при великій кількості робочих місць. На даний момент останньою доступною версією SQL Server є версія 15.0 випуску 2019 року. Для керування базами даних Microsoft SQL Server використовується програмне середовище Microsoft SQL Server Management Studio. З його допомогою можна виконувати маніпуляції з базами даних, у тому числі такі як резервне копіювання і відновлення баз даних.

До переваг можна віднести [6]:

- СУБД масштабується, тому працювати з нею можна на портативних ПК або потужної мультипроцесорної техніці. Процесор може одночасно обробляти великий обсяг запитів;
- Розмір сторінок – до 8 кб, тому дані витягуються швидко, детальну і складну інформацію зберігати зручніше. Система дозволяє обробляти транзакції в інтерактивному режимі, є динамічна блокування;
- Рутинні адміністративні завдання автоматизовані: це управління блокуваннями, пам'яттю, редактура розмірів файлів. У системи продумані настройки, можна створити профілі користувачів;
- Реалізовано пошук по фразах, тексту, словами, можна створювати ключові індекси;

- У SQL Server є реплікації через інтернет, передбачена синхронізація. Є повноцінний веб-асистент для форматування сторінок;
- В систему інтегрований сервер інтерактивного аналізу для прийняття рішень, створення корпоративних звітів. Є служби перетворення інформації;
- Запити можна формулювати англійською мовою, без програмування;
- СУБД підтримує роботу з іншими продуктами Microsoft: Access, MS Excel.

До недоліків відносяться:

- залежність від операційного середовища: СУБД працює тільки з системою Windows;
- висока ціна програми.

4.2 Середовище Microsoft Visual Studio

Microsoft Visual Studio – серія продуктів фірми Майкрософт, які включають в себе інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів [7]. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтрімкою технології Windows Forms, а також веб-сайти, веб-служби, веб-застосунки як в рідному, так і в керованому кодах для всіх платформ, что підтрімуються Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, Windows Phone, .NET Compact Framework та Microsoft Silverlight.

До переваг даного середовища можна віднести:

- Вбудований Web-сервер;
- Підтримка безлічі мов при розробці;
- Менше коду для написання;
- Інтуїтивний стиль кодування;
- Більш висока швидкість розробки;

- Можливості налагодження;
- Можливість управління проектом;
- Вбудована функція управління вихідним кодом;
- Можливість рефакторізації коду;
- Потужна модель розширюваності.

Як недолік можна відзначити неможливість відладчика (Microsoft Visual Studio Debugger) відстежувати в коді режиму ядра. Налагодження в режимі ядра в загальному випадку виконується при використанні WinDbg, KD або SoftICE.

4.3 Середовище Microsoft SQL Server Management Studio

SQL Server Management Studio (SSMS) – утиліта з Microsoft SQL Server 2005 і більш пізніх версій для конфігурації, управління і адміністрування всіх компонентів Microsoft SQL Server [8]. Утиліта включає скриптовий редактор і графічну програму, яка працює з об'єктами і налаштуваннями сервера.

Головним інструментом SQL Server Management Studio є Object Explorer, який дозволяє користувачеві переглядати, отримувати об'єкти сервера, а також повністю ними управляти.

Також є SQL Server Management Studio Express для Express версії сервера. Однак в ній немає підтримки ряду компонентів (Analysis Services, Integration Services, Notification Services, Reporting Services) і SQL Server 2005 Mobile Edition.

4.4 Transact-SQL

Transact-SQL (T-SQL) – процедурне розширення мови SQL, створене компанією Microsoft (для Microsoft SQL Server) і Sybase (для Sybase ASE). Є реалізацією стандарту ANSI/ISO до структурованої мови запитів (SQL) з розширеннями [9]. Використовується для роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства.

SQL був розширений наступними додатковими можливостями:

- керуючі оператори;
- локальні і глобальні змінні;
- різні додаткові функції для обробки рядків, дат, математики і т. п.;
- підтримка аутентифікації Microsoft Windows.

Мова Transact-SQL є ключем до використання MS SQL Server. Всі додатки, які взаємодіють з екземпляром MS SQL Server, незалежно від їх реалізації і призначеного для користувача інтерфейсу, відправляють сервера інструкції Transact—SQL.

4.5 Мова програмування C#

C # – об'єктно-орієнтована мова програмування. Розроблена в 1998-2001 роках групою інженерів компанії Microsoft під керівництвом Андерса Хейлсберг і Скотта Вільтаумота як мова для розробки додатків для платформи Microsoft .NET Framework [10] .

C # відноситься до сім'ї мов з C-подібним синтаксисом, найбільш близьким до C ++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (операторів явного і неявного типу), делегати, атрибути, події, ітератори, властивості, узагальнені типи і методи, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML.

Перейнявши багато від своїх попередників – мов C ++, Delphi, Smalltalk і, особливо, Java – C #, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, C # на відміну від C++ не підтримує множинне успадкування класів, проте допускається множинне спадкування інтерфейсів.

C # розроблювалась як мова програмування прикладного рівня для CLR і, як такий, залежить, перш за все, від можливостей самої CLR. Присутність або відсутність тих чи інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльований в відповідні конструкції CLR.

Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C #; подібної взаємодії слід очікувати і в подальшому. CLR надає C #, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких «класичні» мови програмування позбавлені. Наприклад, прибирання сміття не реалізоване в самому C #, а проводиться CLR для програм, написаних на C #.

4.6 Технологія ADO.NET

ADO.NET являє собою технологію роботи з даними, яка заснована на можливостях платформи .NET Framework. Ця технологія являє собою набір класів, через які ми можемо відправляти запити до баз даних, встановлювати підключення, отримувати відповідь від бази даних і виконувати ряд інших операцій.

Причому важливо зазначити, що систем управління баз даних може бути безліч, причому у своїй сутності вони можуть відрізнятися. Так, наприклад, MS SQL Server для створення запитів використовує мову T-SQL, а MySQL і Oracle застосовують мову PL-SQL [11]. Різні системи баз даних можуть мати різні типи даних. Також можуть відрізнятися якісь інші моменти. Однак функціонал ADO.NET побудований таким чином, щоб надати розробникам уніфікований інтерфейс для роботи з різноманітними СУБД.

Основу інтерфейсу взаємодії з базами даних в ADO.NET представляє обмежене коло об'єктів: Connection, Command, DataReader, DataSet і DataAdapter. За допомогою об'єкта Connection відбувається установка підключення до джерела даних. Об'єкт Command дозволяє виконувати операції з даними з БД. Об'єкт DataReader зчитує отримані в результаті запиту дані. Об'єкт DataSet призначений для зберігання даних з БД і дозволяє працювати з ними незалежно від БД. І об'єкт DataAdapter є посередником між DataSet і джерелом даних. Головним чином, через ці об'єкти і буде йти робота з базою даних.

Однак щоб використовувати один і той же набір об'єктів для різних джерел даних, необхідний відповідний провайдер даних. Власне через провайдер даних в ADO.NET і здійснюється взаємодія з базою даних. Причому для кожного

джерела даних в ADO.NET може бути свій провайдер, який власне і визначає конкретну реалізацію вищевказаних класів.

За замовчуванням в ADO.NET є наступні вбудовані провайдери:

- Провайдер для MS SQL Server;
- Провайдер для OLE DB (Надає доступ до деяких старих версій MS SQL Server, а також до БД Access, DB2, MySQL і Oracle);
- Провайдер для ODBC (провайдер для тих джерел даних, для яких немає своїх провайдерів);
- Провайдер для Oracle;
- Провайдер EntityClient. Провайдер даних для технології ORM Entity Framework;
- Провайдер для сервера SQL Server Compact 4.0.

Схематично архітектуру ADO.NET можна представити таким чином:

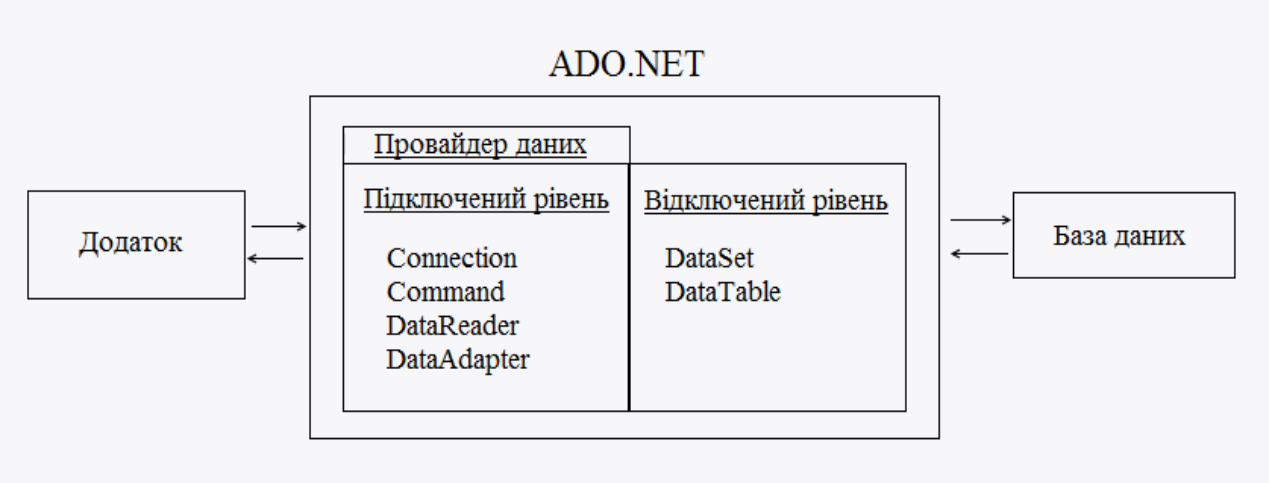

Рисунок 4.1 – Архітектура ADO.NET

Функціонально класи ADO.NET можна розбити на два рівня: підключений і відключений. Кожен провайдер даних .NET реалізує свої версії об'єктів Connection, Command, DataReader, DataAdapter і ряду інших, який складають підключений рівень. Тобто за допомогою них встановлюється підключення до БД і виконується з нею взаємодія.

5 ІНТЕРФЕЙС КОРИСТУВАЧА

Користувацький інтерфейс включає у себе одну панель, на якій відбуваються усі події (рис. 5.1). Дана панель включає у себе: консоль подій, поле вводу запитів та об’єкти взаємодії з базою даних.

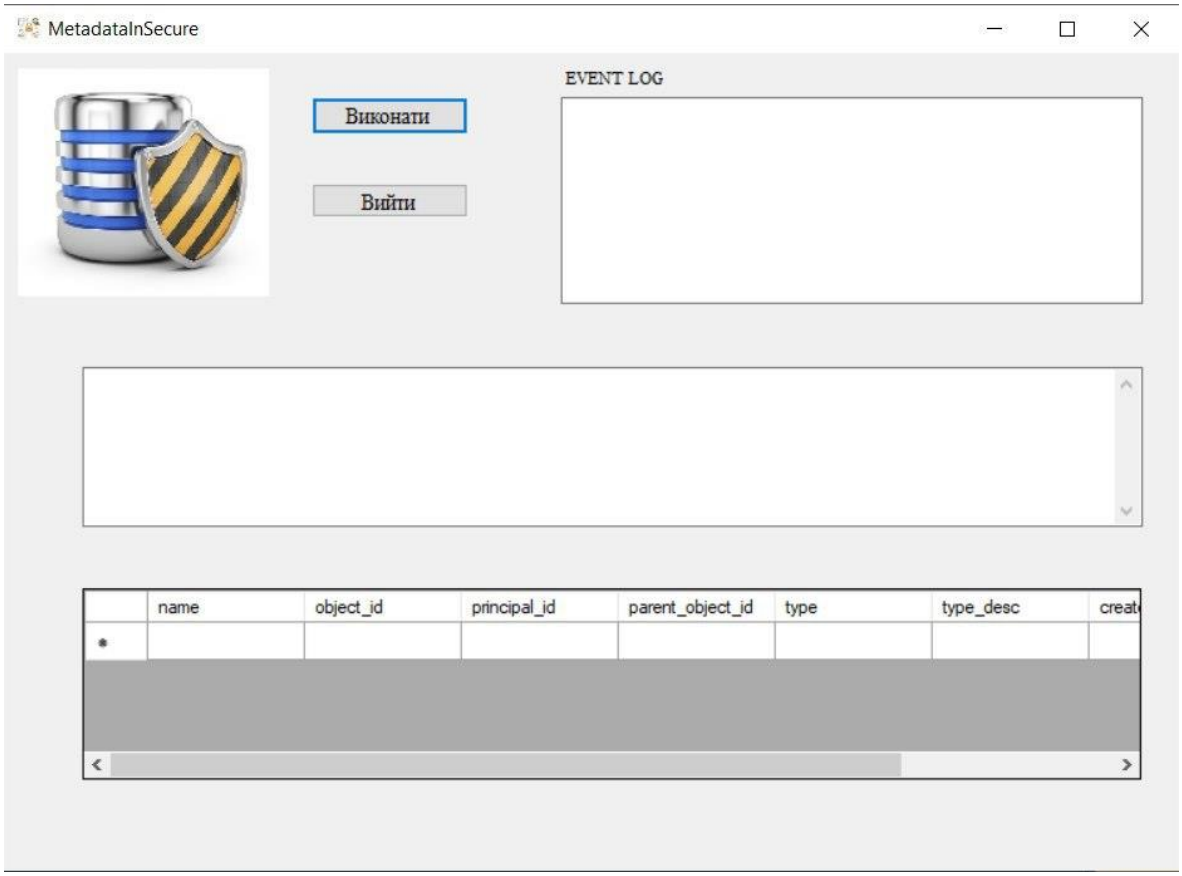

Рисунок 5.1 – Інтерфейс додатку

Консоль (Event log) записує усі зміни у системі та фіксує помилки. З її допомогою можна зручно відслідковувати події системи та знаходити помилку роботи, оскільки усі події відслідковуються у режимі реального часу. Можливості консолі дуже добре показані на рисунку 5.2.

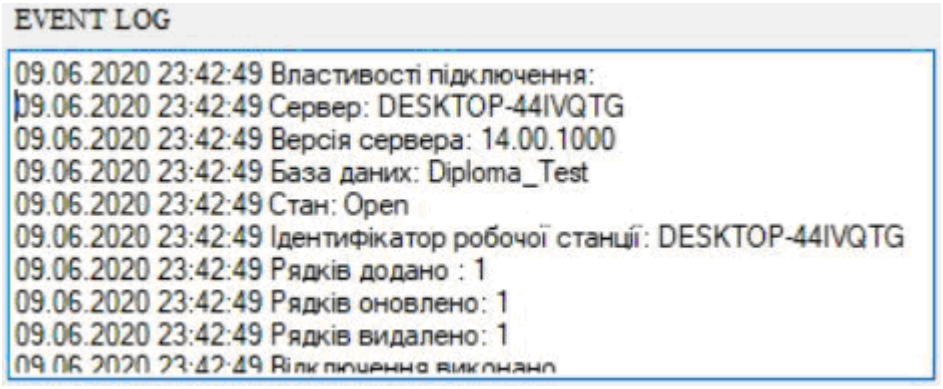

Рисунок 5.2 – Відображення подій у консолі

Поле вводу запитів являє собою простір для вводу та зчитування запитів до бази даних. Запити розділяються за допомогою символа ‘;’. Після вводу запитів користувач має натиснути клавішу “Виконати”, після чого запити будуть відправлені до бази даних сервера MS SQL і будуть виведені результати (рис. 5.3).

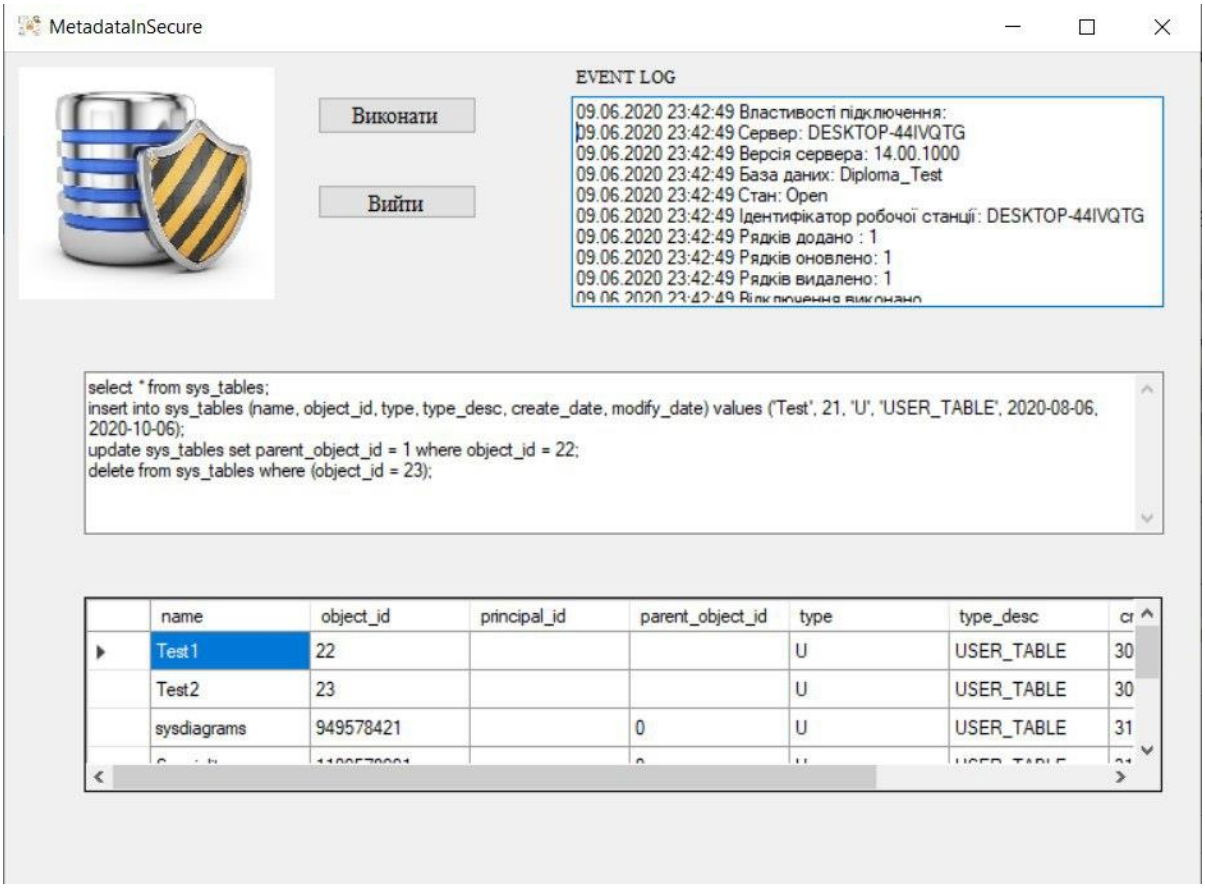

Рисунок 5.3 – Приклад виконання серії запитів

Якщо, з якоїсь причини, поле вводу виявилося порожнім, а користувач натиснув клавішу “Виконати”, то додаток видасть відповідне повідомлення про помилку, як показано на рисунку 5.4.

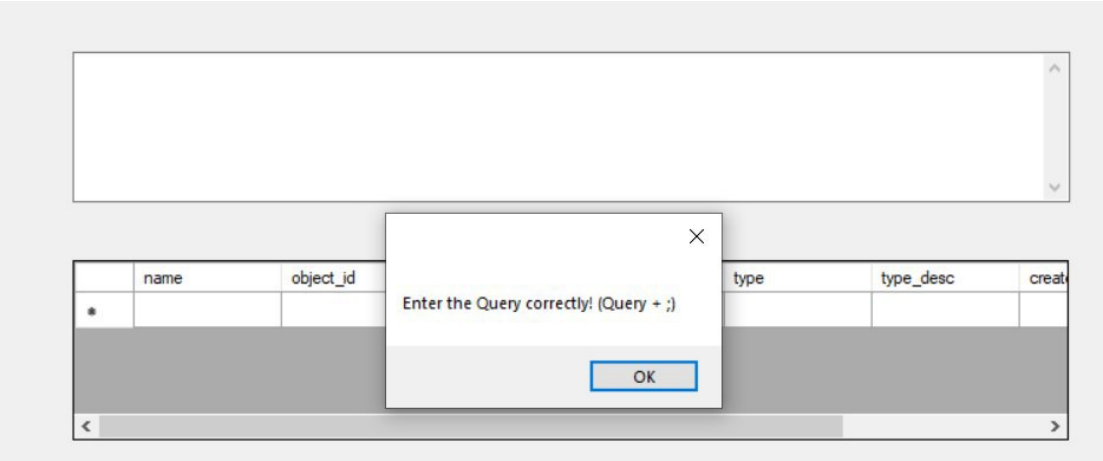

Рисунок 5.4 – Помилка порожнього рядка вводу

Якщо ж запит було введено, проте він містить у собі некоректні параметри або помилки у структурі самого запиту, тоді буде отримано відповідне попередження у вигляді повідомлення та у консолі (рис. 5.5).

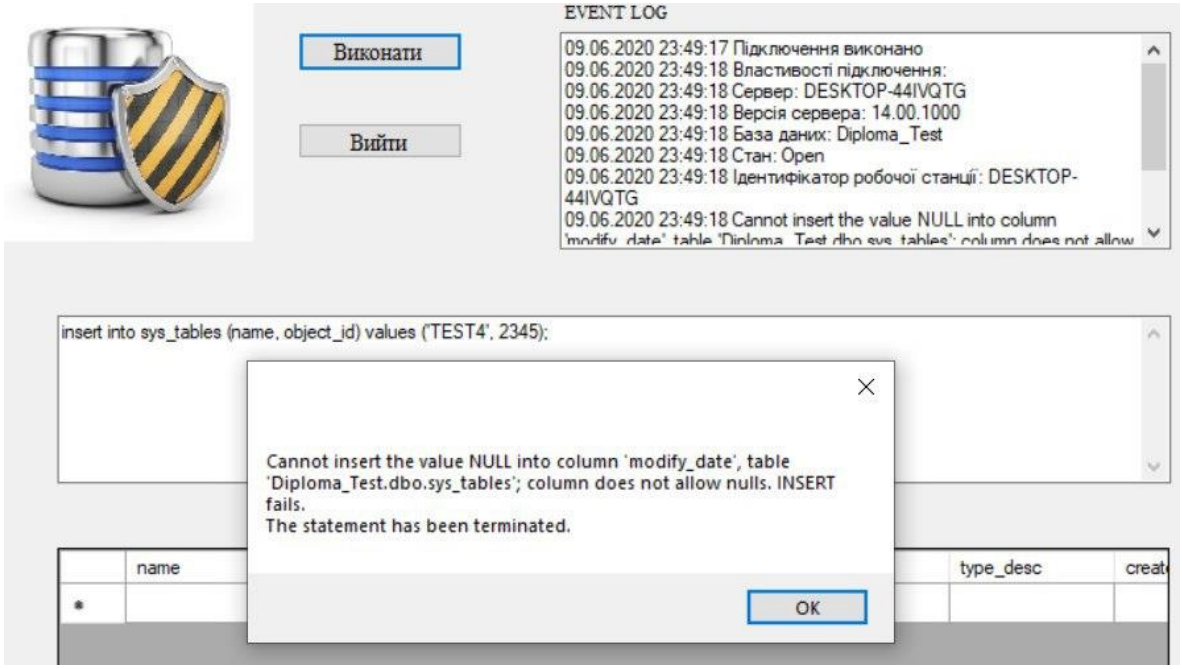

Рисунок 5.5 – Помилка синтаксису SQL-запиту

Після завершення роботи з додатком і подальшим натисканням клавіші “Вийти”, усі події даного сеансу роботи будуть вписані до текстового документу і збережені у ньому (рис. 5.6).

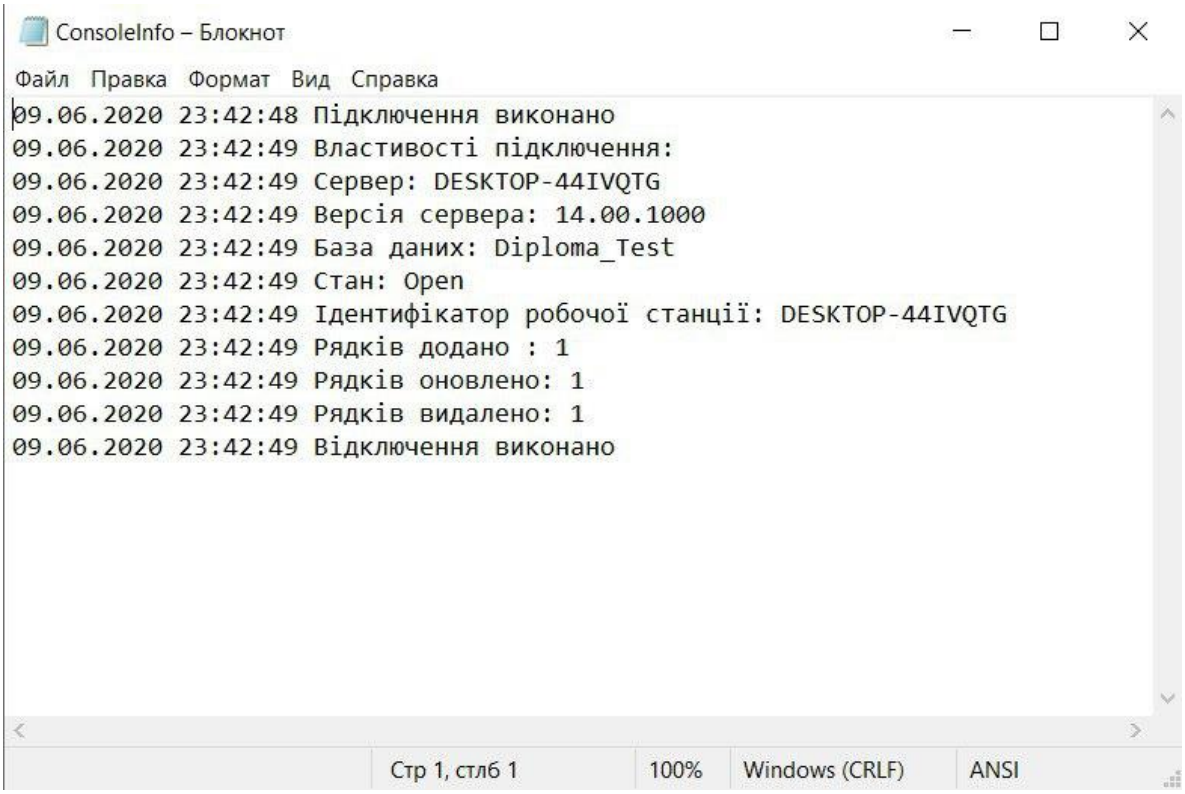

Рисунок 5.6 – Відображення усіх подій у файлі

Вся суть роботи даного додатку зводиться до недопущення потенційного користувача до прямої взаємодії з метаданими бази. Використовуючи даний додаток, користувач (з відповідними дозволами) зможе працювати з метаданими без ризику для усієї системи, оскільки він унеможливлює виникнення несправностей чи знищення цілої бази даних через необережність користувача.

Оскільки даний продукт створювався для роботи з MS SQL SERVER, то його можна туди інтегрувати. Для впровадження в середовище MS SQL саме розробленого компоненту, необхідно провести наступні маніпуляції у середовищі Microsoft SQL Server Management Studio з використанням Transact-SQL:

1. Проводимо інтеграцію збірки до середовища використання:

a. Встановлюємо з'єднання з компонентом Компонент Database Engine.
b. На панелі “Стандартна” натискаємо “Створити запит”.
c. Вводимо наступний текст у вікно запиту і натискаємо кнопку “Виконати”, де вказуємо назву збірки, шлях до розташування збірки на пристрої та сет дозволів (рис. 5.7).

```
CREATE ASSEMBLY MetadataInSecure
    FROM 'D:\Projects\MetadataInSafety\bin\Debug\MetadataInSecure'
    WITH PERMISSION_SET = SAFE
```

Рисунок 5.7 – Створення збірки на Microsoft SQL Server

d. Для створення процедури вводимо наступний текст у вікно запиту і натискаємо кнопку “Виконати” (рис.5.8). Замість значення <procedure_name>, вказуємо їм’я збереженої процедури.

```
CREATE PROCEDURE <prodecure_name>
    AS EXTERNAL NAME MetadataInSecure.StoredProcedures
```

Рисунок 5.8 – Створення збереженої процедури на SQL Server

2. За необхідності можемо встановити процедуру в автоматичний режим виконання:
   a. На панелі “Стандартна” натискаємо “Створити запит”.
   b. Вводимо наступний текст у вікно запиту і натискаємо кнопку “Виконати” (рис. 5.9). У значення параметра @ProcName підставляємо ім’я збереженої процедури, яке ми надали у пункті 1.

```
EXEC sp_procoption @ProcName = '<procedure name>';
     , @OptionName = 'startup';
     , @OptionValue = 'on';
```

Рисунок 5.9 – Встановлення збереженої процедури на автоматичне виконання

c. Для скасування автоматичного виконання процедури вводимо наступний текст у вікно запиту і натискаємо кнопку “Виконати” (рис. 5.10). У значення параметра @ProcName підставляєте ім’я збереженої процедури, яке ви надали у пункті 1.

```
EXEC sp_procoption @ProcName = '<procedure name>';
     , @OptionValue = 'off';
```

Рисунок 5.10 – Скасування автоматичного виконання збереженої процедури

Ось і все. Збережена процедура успішно інстальована на ваш SQL Server. Користуйтесь грамотно і бережіть дані.

ВИСНОВКИ

У ході аналізу існуючого програмного забезпечення безпеки метаданих було досліджено системи, які використовуються для вирішення подібних задач. Аналіз показав, що існуючі системи вирішують задачу далеко не у повному обсязі, оскільки даною проблемою майже не опікуються.

Розроблений програмний продукт є актуальним, оскільки розробляється як частина більшої системи. Розроблена система дозволяє легко модернізувати себе додатковим забезпеченням, тобто має потенціал до розвитку. Це дає змогу підвищити гнучкість та зручність системи, як у розробці та супроводі, так і у використані.

Проведено огляд методів і засобів розробки програмної системи. Отримано представлення про ситуацію у сфері безпеки даних на момент сьогодення, а отже проаналізовані сильні та слабкі сторони сучасних механізмів забезпечення безпеки даних. За результатами виконання тестових завдань була підтверджена коректність отриманих результатів, а отже система відповідає поставленим вимогам.

Користувачами системи можуть бути різноманітні користувачі бази даних, які мають на це відповідний дозвіл від адміністратора.

Робота над дипломним проектом покращила знання різноманітних технологій, що використовуються під час розробки програмного забезпечення. Також було проведено дослідження різноманітних технік та алгоритмів машинного навчання, було виявлено сфери, для яких можна застосовувати дану технологію. Також було створено декілька прототипів програмного забезпечення, які вирішували різноманітні аспекти поставленої задачі, а також які лягли в основу розробленого програмного забезпечення.

Програмна система має клаєнт-серверну архітектуру. Для реалізаціїї системи використане різноманітне програмне забезпечення: Microsoft SQL Server Management Studio, Visual Studio 2017, Transact-SQL та інші.

ДОДАТОК 1

Додаткові засоби захисту бази метаданих реєстру інформаційних ресурсів

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР61115_20Б

Аркушів 2

Київ – 2020

Позначення	Найменування	Примітки
	Документація	
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР61115_20Б 81-1	Записка.docx	Текстова частина дипломної роботи
	Компоненти	
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР61115_20Б 12-1		Текс програмного продукту
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР61115_20Б 12-2		Опис програмного модулю

ДОДАТОК 2

Додаткові засоби захисту бази метаданих реєстру інформаційних ресурсів

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР61115_20Б

Аркушів 4

Київ – 2020

Код інтерфейс додатку

```
public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу "diploma_TestDataSet1.sys_tables". При необходимости она может быть перемещена или удалена.
            //this.sys_tablesTableAdapter.Fill(this.diploma_TestDataSet1.sys_tables);

        }

        private void execute_Click(object sender, EventArgs e)
        {
            string connectionString = "Data Source=DESKTOP-44IVQTG;Initial Catalog=Diploma_Test;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnet Failover=False";
            string[] Query = textBox1.Text.Split(';');
            int Query_Length = Query.Length;

            //перевірка на порожній рядок вводу
            if (Query_Length == 1)
            {
                MessageBox.Show("Enter the Query correctly! (Query + ;)");
            }

            else
            {
                //відкриття підключення
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    try
                    {
                        connection.Open();
                        textBox2.Text = "";
                        textBox2.Text += DateTime.Now + " " + "Підключення виконано" + System.Environment.NewLine;
                        System.Threading.Thread.Sleep(1000);

                        //вивід інформації про підключення
                        textBox2.Text += DateTime.Now + " " + "Властивості підключення:" + System.Environment.NewLine;
                        textBox2.Text += DateTime.Now + " " + "Сервер: " + connection.DataSource + System.Environment.NewLine;
```

```
                    textBox2.Text += DateTime.Now + " " + "Версія сервера: " +
connection.ServerVersion + System.Environment.NewLine;
                    textBox2.Text += DateTime.Now + " " + "База даних: " +
connection.Database + System.Environment.NewLine;
                    textBox2.Text += DateTime.Now + " " + "Стан: " + connection.State +
System.Environment.NewLine;
                    textBox2.Text += DateTime.Now + " " + "Ідентифікатор робочої станції: " +
connection.WorkstationId + System.Environment.NewLine;
foreach (string sqlExpression in Query)
                {
                    if (sqlExpression.Contains("insert"))
                    {
                        SqlCommand command = new SqlCommand(sqlExpression, connection);
                        int number = command.ExecuteNonQuery();
                        textBox2.Text += DateTime.Now + " " + "Рядків додано : " + number +
System.Environment.NewLine;
                    }
                    if (sqlExpression.Contains("select"))
                    {
                        SqlCommand select = new SqlCommand(sqlExpression, connection);
                        SqlDataReader reader = select.ExecuteReader();
                        //string sql = reader;
                        //sys_tablesTableAdapter = new SqlDataAdapter(sql, connection);
                        this.sys_tablesTableAdapter.Fill(this.diploma_TestDataSet1.sys_tables);
                        reader.Close();
                    }
                    if (sqlExpression.Contains("update"))
                    {
                        SqlCommand command = new SqlCommand(sqlExpression, connection);
                        int number2 = command.ExecuteNonQuery();
                        textBox2.Text += DateTime.Now + " " + "Рядків оновлено: " + number2 +
System.Environment.NewLine;
                    }
                    if (sqlExpression.Contains("delete"))
                    {
                        SqlCommand command = new SqlCommand(sqlExpression, connection);
                        int number3 = command.ExecuteNonQuery();
                        textBox2.Text += DateTime.Now + " " + "Рядків видалено: " + number3 +
System.Environment.NewLine;
                    }
                    Console.WriteLine(sqlExpression);
                }

            }
            catch (SqlException ex)
            {
                textBox2.Text += DateTime.Now + " " + ex.Message +
System.Environment.NewLine;
                MessageBox.Show(ex.Message);
            }
            finally
            {
```

```
                //закриття підключення
                textBox2.Text += DateTime.Now + " " + "Відключення виконано" +
System.Environment.NewLine;

                string WritePath =
@"D:\Диплом\MetadataInSecure_2.0\MetadataInSecure_2.0\obj\Debug\ConsoleInfo.txt";

                using (StreamWriter sw = new StreamWriter(WritePath, false,
System.Text.Encoding.Default))
                {
                    sw.WriteLine(textBox2.Text);
                }
            }
        }
    }

    private void textBox1_TextChanged(object sender, EventArgs e)
    {
    }

    private void exit_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }
}
```

Код форми додатку

```
partial class Form1
    {
        /// <summary>
        /// Обязательная переменная конструктора.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        /// <param name="disposing">истинно, если управляемый ресурс должен быть удален;
иначе ложно.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
private System.Windows.Forms.Button execute;
```

```
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.DataGridView dataGridView1;
        private Diploma_TestDataSet diploma_TestDataSet;
        private System.Windows.Forms.BindingSource studentsBindingSource;
        private Diploma_TestDataSetTableAdapters.StudentsTableAdapter studentsTableAdapter;
        private System.Windows.Forms.Button Exit;
        private System.Windows.Forms.BindingSource studentsBindingSource1;
        private Diploma_TestDataSet1 diploma_TestDataSet1;
        private System.Windows.Forms.BindingSource systablesBindingSource;
        private Diploma_TestDataSet1TableAdapters.sys_tablesTableAdapter sys_tablesTableAdapter;
        private System.Windows.Forms.DataGridViewTextBoxColumn
nameDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
objectidDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
principalidDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
parentobjectidDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
typeDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
typedescDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
createdateDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
modifydateDataGridViewTextBoxColumn;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.Label label1;
    }
}
```

ДОДАТОК З

Додаткові засоби захисту бази метаданих реєстру інформаційних ресурсів

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР61115_20Б

Аркушів 5

Київ – 2020

АНОТАЦІЯ

Розроблений програмний продукт забезпечує захист цілісності та конфіденційності метаданих у базі даних MS SQL Server.

Дипломна робота присвячена розробці програмного додатку на базі Microsoft SQL Server засобами платформи Windows Forms, Microsoft SQL Server Management Studio, Visual Studio 2017 для подальшого використання у даній системі.

ЗМІСТ

1 ВІДОМОСТІ ПРО ПРОГРАМНИЙ МОДУЛЬ

Клієнт-додаток називається MetadataInSecure. Даний продукт розорблено у середовищі Visual Studio 2017, використовуючи об'єктно-орієнтовану мову програмування C#, фреймворк .NET, та Windows Forms.

Програма призначена для забезпечення цілісності та недоторканості метаданих бази на сервері.

1.1 Опис логічної структури

Програмний продукт було розроблено у вигляді декстопного додатку за допомогою інструментів Windows Forms.

Данна система складається з основного меню, де виконуються усі дії.

Консольний додаток отримує, аналізує інформацію та має окремий блок для обміну інформацією з базою даних та виводу подій.

У разі вдалої роботи програми, користувач буде здатний отримати доступ до певних даних (згідно з його дозволами).

1.2 Вхідні та вихідні дані

Вхідними даними є SQL-запити до бази даних на сервері Microsoft SQL Server.

Вихідними даними є список подій (можливих помилок) та результати виконаних запитів.

ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Програмний модуль було протестовано на комп’ютері, який побудовано на базі процесору архітектури x64 Intel Core i3 та має 8 Гб оперативної пам’яті. Розроблене програмне забезпечення не потребує значних апаратних витрат, тому може бути використане на менш потужних комп’ютерах, оскільки всі дії відбуваються у додатку.